Tomato Tracking Assignment Report

Author: Nada Abbas

Date: August 23, 2024

Contents

1 Executive Summary

This report presents a detailed account of the development of a robust system designed for the automatic detection, classification, and tracking of tomatoes. Initially, the project considered using a Mask R-CNN model, but due to dataset limitations, YOLOv5 was chosen as a more suitable alternative. YOLOv5's balance between speed and accuracy made it ideal for this task, particularly given the data constraints. The system aims to enhance agricultural automation, specifically improving the efficiency of monitoring and harvesting processes. This report covers the methodology, key results, and insights gained, along with recommendations for future enhancements.

2 Introduction

This report details the development of a system that automatically detects, classifies, and tracks tomatoes, addressing challenges such as varying stages of ripeness and sizes. Initially, Mask R-CNN was considered, but after evaluating the dataset, YOLOv5 was selected for its superior performance in scenarios with limited data. YOLOv5 is optimized for real-time object detection, offering a balance between speed and accuracy, which is crucial for agricultural applications like automated harvesting and quality control. Its architecture allows for rapid inference, making it suitable for environments requiring quick, reliable decisions. Moreover, YOLOv5's robust performance with smaller datasets, enhanced by advanced data augmentation techniques like mosaic augmentation, ensures its effectiveness in real-world scenarios. Additionally, its efficient use of computational resources makes it ideal for deployment on common agricultural hardware, such as edge devices or drones.

3 Methodology

3.1 Data Collection and Preprocessing

Initially, the plan was to employ a Mask R-CNN model for tomato detection. However, due to limitations in the size of the available dataset, a strategic decision was made to switch to the YOLOv5 model, which is more suited to the data volume. The dataset underwent preprocessing steps, including resizing, normalization, and augmentation, to enhance model performance and robustness. The preprocessing steps are encapsulated in the Data Preprocessing module, as illustrated in the UML diagram, which handles the conversion of dataset annotations into the YOLO format and processes the data for model training.

3.2 Model Development

The YOLOv5 model was selected for its superior speed and accuracy in object detection tasks, making it ideal for real-time applications. This model's architecture, which includes components such as CSPDarknet53 for the backbone and a YOLOv3 head, allows it to efficiently handle the simultaneous detection and classification of tomatoes, even in scenarios involving occlusions and varying lighting conditions. The UML diagram represents this as the YOLOv5Model class, detailing how the model architecture is structured and how the training and detection processes are implemented.

3.3 Tracking Algorithm

Post-training, the DeepSort algorithm was integrated with YOLOv5 to facilitate the tracking of tomatoes across consecutive frames. This integration ensures that each tomato is uniquely tracked and counted, avoiding the issue of double-counting. In the UML diagram, this process is depicted by the DeepSort module, which interacts with the YOLOv5Model to process tracking data. DeepSort's robust tracking capabilities, combined with YOLOv5's detection accuracy, result in a system that maintains precise and consistent tracking performance.

3.4 System Overview

The UML diagram (Figure 1) provides a comprehensive view of the system's architecture, highlighting the interactions between the Data Preprocessing, YOLOv5Model, and DeepSort modules. This structured representation of the system aids in understanding how each component contributes to the overall functionality, from data preparation to detection and tracking, ensuring a cohesive and well-coordinated approach to solving the problem of tomato detection and tracking.

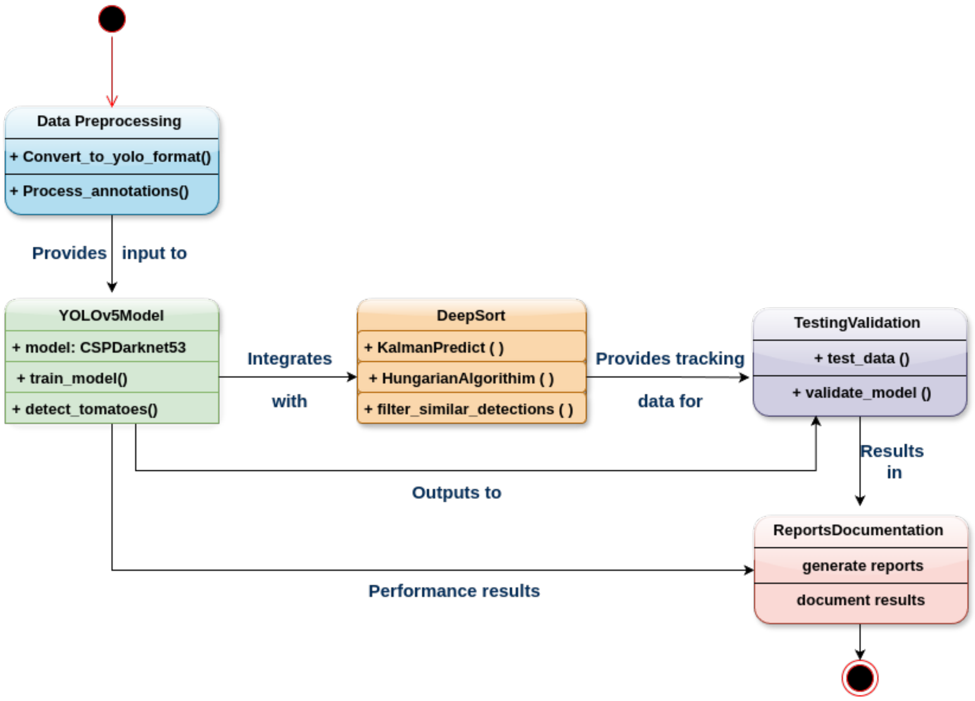

Figure 1: UML Diagram of the Tomato Tracking System. The diagram illustrates the interactions between the Data Preprocessing, YOLOv5Model, DeepSort, and Testing/Validation modules, as well as the flow of data and performance results through the system.

4 Results & discussion

4.1 Training the Model

The YOLOv5 model underwent extensive training to achieve optimal accuracy in detecting and classifying tomatoes. The training process involved fine-tuning several hyperparameters, adjusting the learning rate, and employing data augmentation techniques to enhance the model's robustness. The table below summarizes the key performance metrics obtained during the training phase:

Metric	**Precision**	**Recall**	**mAP@50**	**mAP@50-95**
Value	83.8%	74.2%	84.0%	69.1%

Table 1: Summary of key performance metrics achieved during training.

These metrics highlight the model's strong capability to accurately detect and classify tomatoes across multiple categories, including fully ripened, half-ripened, and green tomatoes of different sizes (normal and cherry). The balance between precision and recall indicates that the model is both accurate in its predictions and comprehensive in identifying relevant instances.

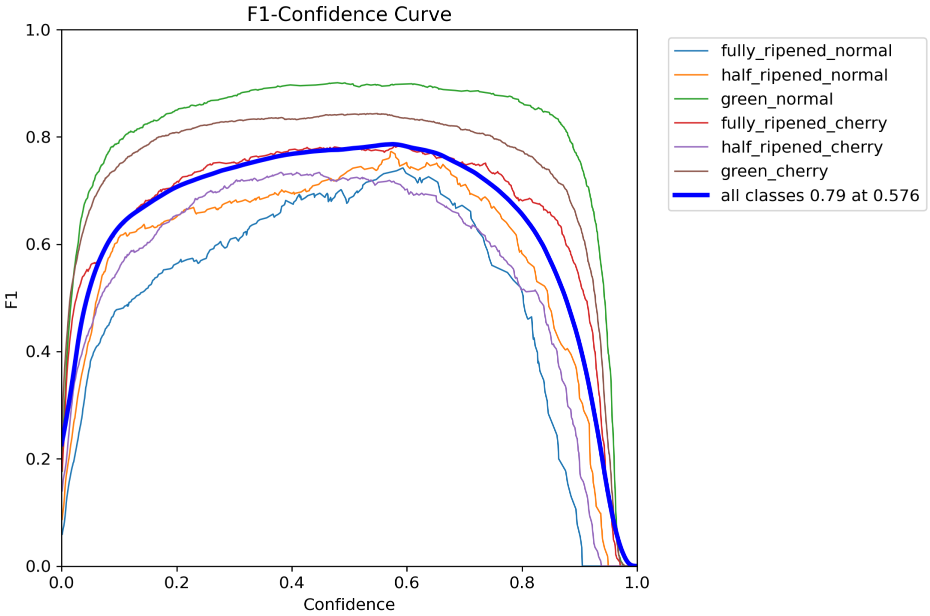

Figure 2: F1-Confidence Curve showing the variation of the F1 score with different confidence thresholds. The peak at a confidence level of 0.576 indicates optimal performance.

The F1-Confidence curve illustrates the model's ability to balance precision and recall across different confidence thresholds, with an optimal performance observed at a confidence level of 0.576. This indicates that the chosen threshold maximizes the model's F1 score, ensuring a good trade-off between precision and recall.

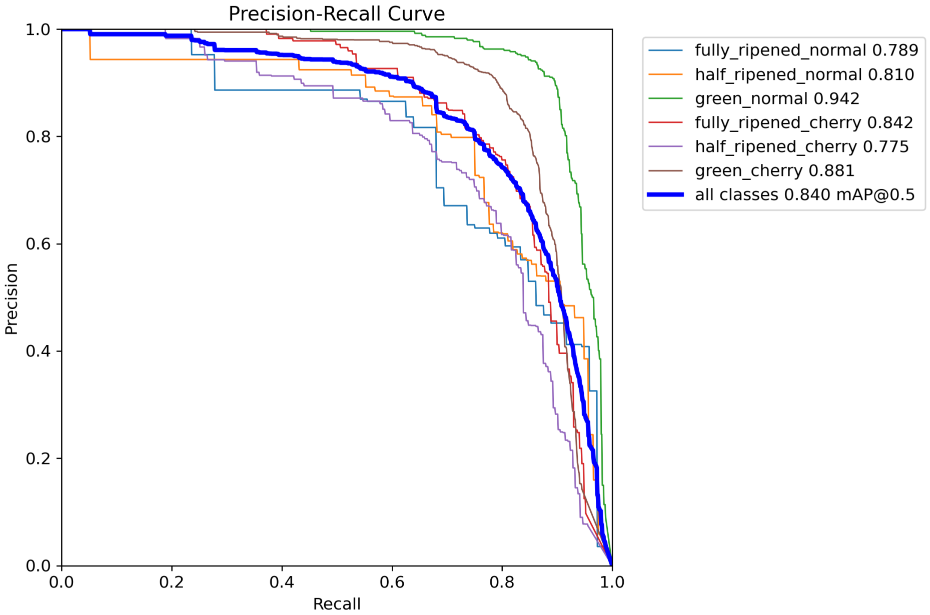

Figure 3: The model achieves a mAP@50 of 84.0%, with particularly high performance

The Precision-Recall curve further supports the model's balanced performance, confirming its ability to maintain high precision across varying levels of recall. Notably, the green_normal class demonstrates excellent precision, underscoring the model's effectiveness in distinguishing this particular category of tomatoes.

4.1.1 Training and Validation Losses

The loss curves for training and validation show a consistent downward trend, which indicates that the model is effectively learning and generalizing to new data. The convergence of these curves suggests minimal risk of overfitting, underscoring the model's robustness. The training results confirm the model's robustness and capability in accurately detecting and classifying tomatoes. The high mAP@50 and precision metrics reflect the model's suitability for real-world agricultural applications, where precision and accuracy are paramount.

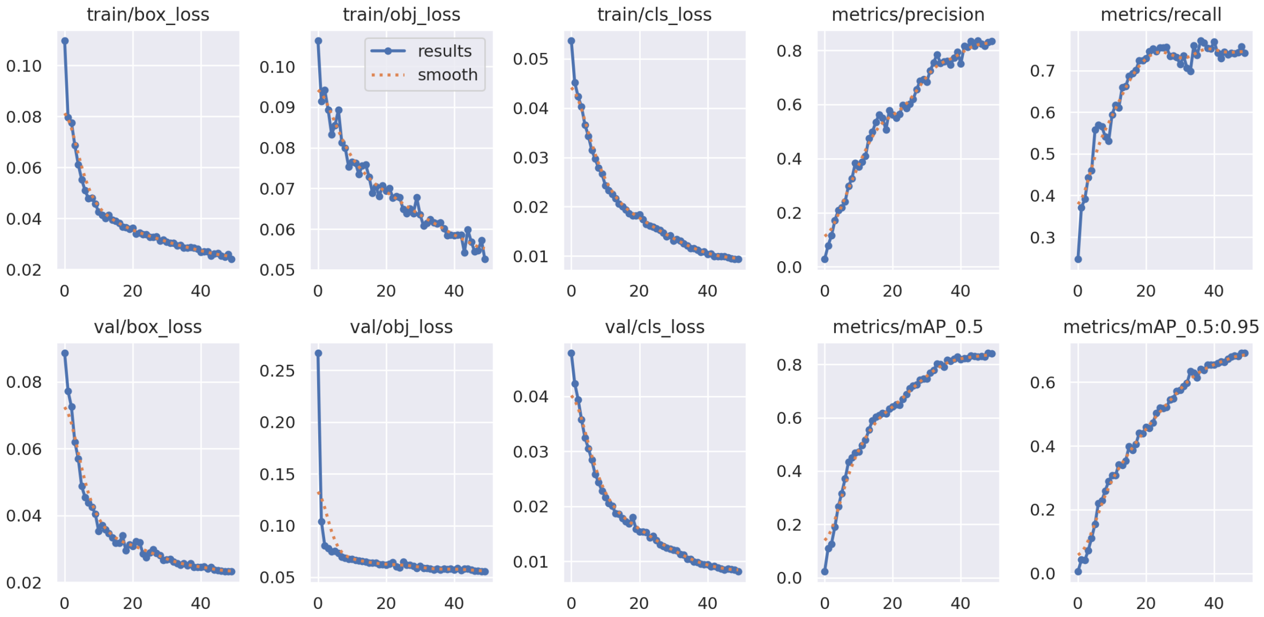

Figure 4: Training and validation loss curves showing a steady decrease over time, indicating effective learning and good generalization of the model.

4.2 Detection Performance

The detection phase evaluated the model's ability to identify and classify tomatoes under various conditions, including differing lighting, occlusions, and tomato appearances. The model's robustness was confirmed through testing on a diverse set of images, resulting in an Overall Detection Accuracy of 85.3%, False Positives of 12.5%, and False Negatives of 9.4%. These results demonstrate that the model performs reliably in general detection tasks, achieving a high accuracy rate with relatively low false positive and negative rates.

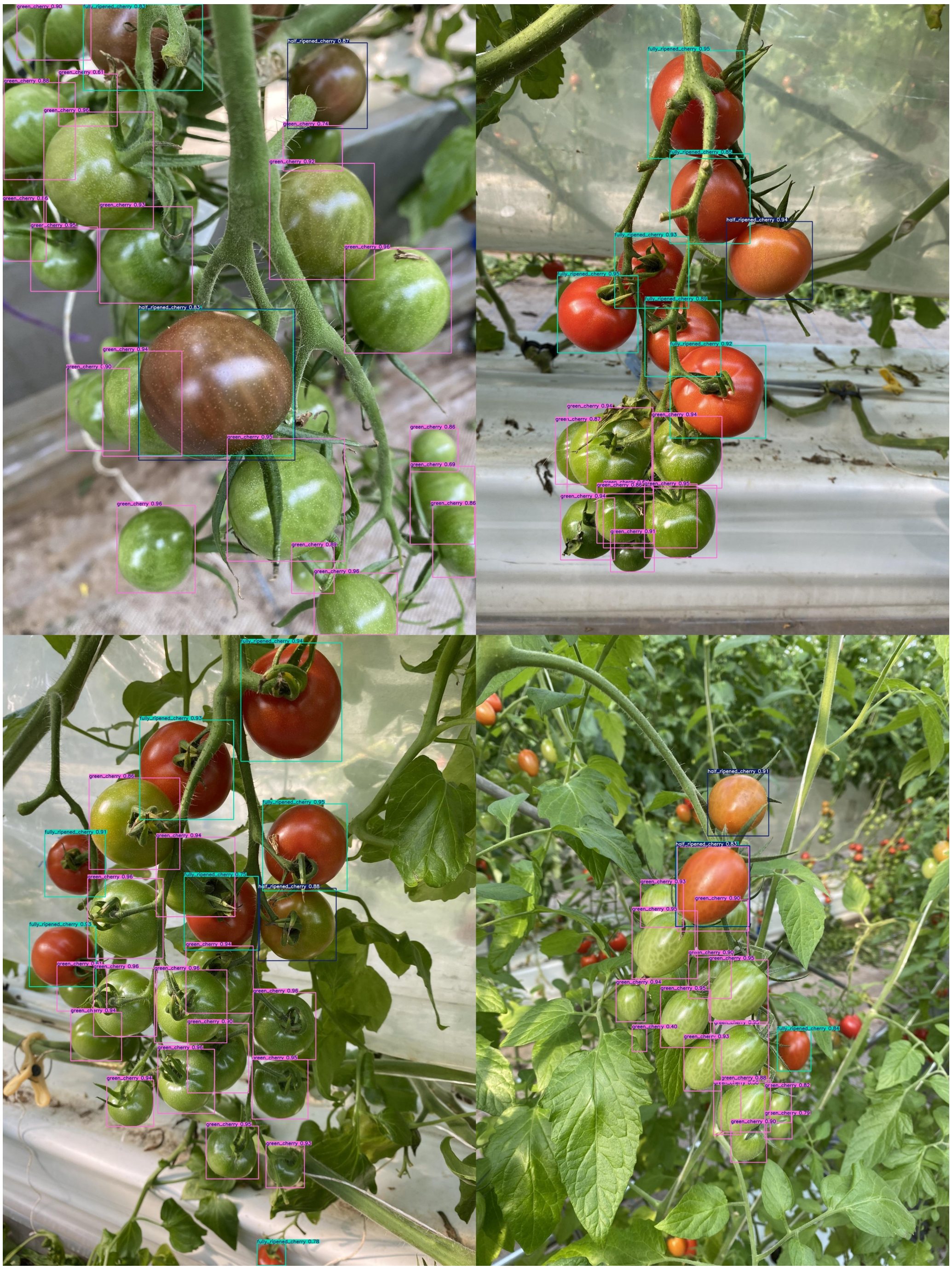

Figure 5: Detection and classification of different classes of tomatoes.

The model was also tested for its ability to differentiate between various tomato classes based on size and ripeness. The results further affirm the model's capability to not only detect but also accurately count tomatoes across different categories, making it a valuable tool for agricultural applications.

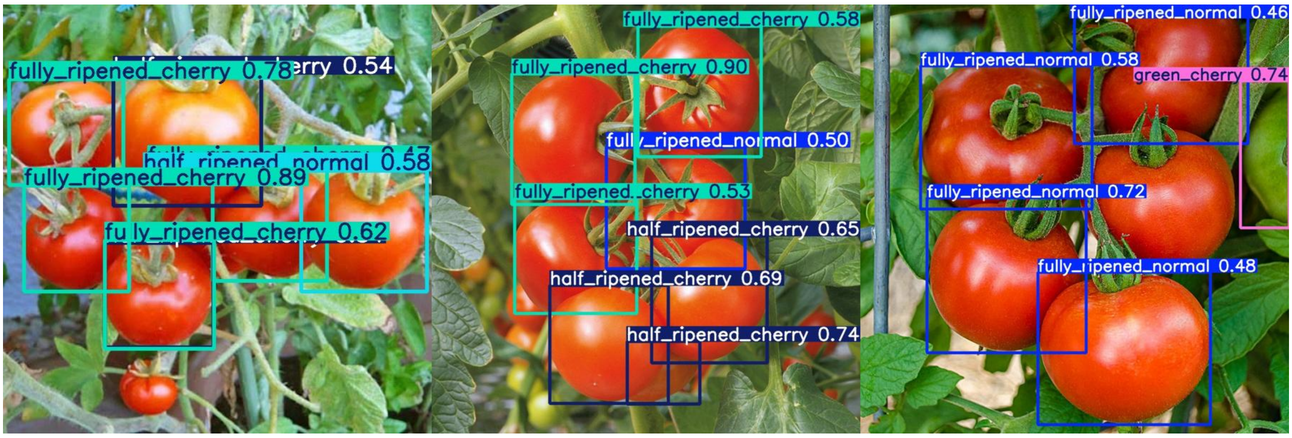
fully_ripened_cherry 0.78
erry 0.54
half_ripened_normal 0.58
fully_ripened_cherry 0.89
fully_ripened_cherry 0.62
fully_ripened_cherry 0.58
fully_ripened_cherry 0.90
fully_ripened_normal 0.50
fully_ripened_cherry 0.53
half_ripened_cherry 0.65
half_ripened_cherry 0.69
half_ripened_cherry 0.74
fully_ripened_normal 0.46
fully_ripened_normal 0.58
green_cherry 0.74
fully_ripened_normal 0.72
fully_ripened_normal 0.48

Figure 6: General detection of tomatoes in various conditions.

4.3 Tracking Across Consecutive Frames

The integration of DeepSort with YOLOv5 ensures reliable tracking and counting of tomatoes across consecutive frames, preventing duplicate counts. However, it was observed that the bounding boxes retrieved after tracking occasionally differ from the initial detection boxes. This discrepancy primarily arises due to the internal processes within DeepSort, particularly the application of the Kalman filter.

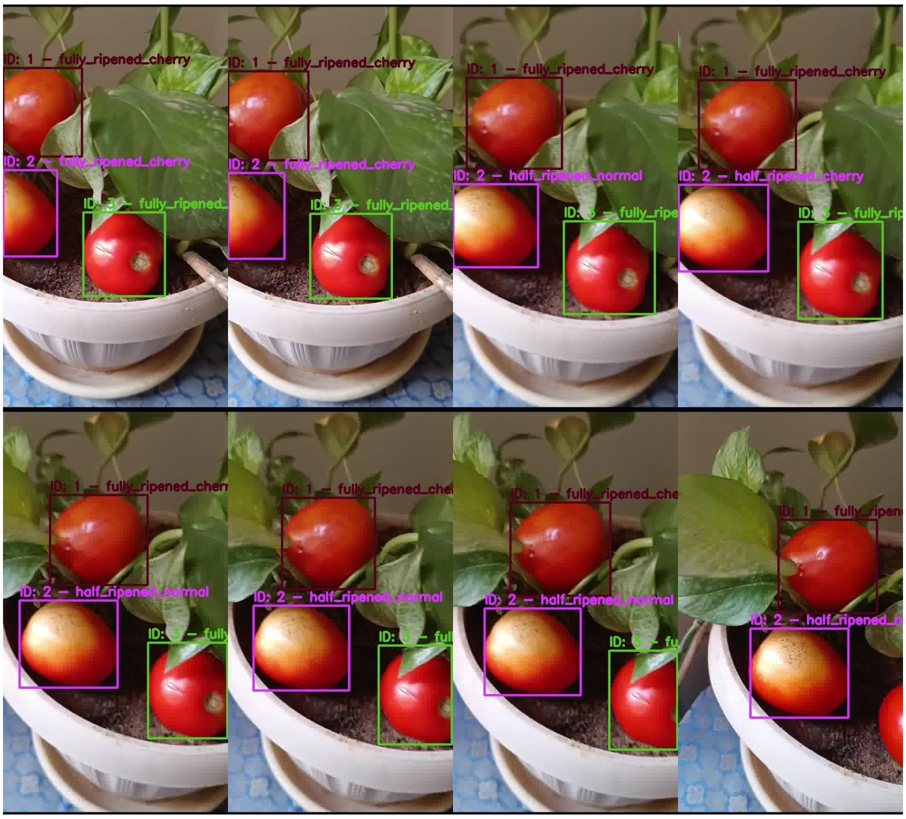
ID: 1 – fully_ripened_cherry
ID: 2 – fully_ripened_cherry
ID: 3 – fully_ripened_
ID: 2 – half_ripened_normal
ID: 2 – half_ripened_cherry
ID: 3 – fully_ripe

Figure 7: Tracking and counting of tomatoes across consecutive frames using DeepSort.

The Kalman filter predicts the position of each tracked object in subsequent frames, considering the object's previous velocity and trajectory. This predictive capability helps maintain smooth tracking, even when objects are partially occluded or moving rapidly. Furthermore, DeepSort's internal state representation, which includes parameters such as bounding box center, aspect ratio, and height, is subject to temporal adjustments. These adjustments are necessary to ensure consistent tracking over time, even though they might introduce slight variations in the bounding box coordinates. Such behavior is expected and essential for maintaining accurate and continuous tracking across multiple frames.

Despite these minor discrepancies, the overall tracking performance remains robust, with DeepSort successfully minimizing issues such as double counting. This functionality is crucial for applications where accurate object counting over time is necessary, such as in automated agricultural systems.

5 Conclusion

The developed system showcases significant potential in automating the detection, classification, and tracking of tomatoes in agricultural settings. The combination of YOLOv5 for detection and DeepSort for tracking provides a reliable solution that ensures high accuracy and consistency across frames. This system can significantly enhance the efficiency of monitoring and harvesting processes, offering a scalable solution for agricultural automation.

Future improvements could focus on expanding the dataset to include a wider variety of tomato species and conditions, further enhancing the model's robustness. Additionally, fine-tuning the Kalman filter and exploring alternative tracking algorithms could help reduce any remaining discrepancies in the tracking process, thereby improving overall system performance.

Contact Information:

If you have any questions or require further information, please contact:

Email: n.elsayed@nu.edu.eg

Appendices

Appendix A: Full Code

The full code for this project is available on GitHub at `https://github.com/NadaAbbas444/Tomato_Detection_Counting_Tracking_yolo`.